LAB II: Securing The Data Path and Routing Infrastructure

8. Create Packet Filters

a. Create a packet filter which will deny packets that have obviously bogus IP source addresses but permit everything else

```
ip access-list extended TestA
deny   ip 127.0.0.0 0.255.255.255 any log
deny   ip 0.0.0.0 0.255.255.255 any log
deny   ip 10.0.0.0 0.255.255.255 any log
deny   ip 169.254.0.0 0.0.255.255 any log
deny   ip 172.16.0.0 0.15.255.255 any log
deny   ip 192.168.0.0 0.0.255.255 any log
deny   ip 224.0.0.0 15.255.255.255 any log
permit ip any any
```

b. Create a packet filter which will allow packets with source IP addresses from your specific network but deny everything else. Use the network that is found under the bgp address-family ipv4 configuration (i.e. for router 14 it would be network 100.4.48.0 mask 255.255.240.0)

```
ip access-list extended TestB
permit <my-network> <mask>
deny any any log
```

c. Create a packet filter which will allow packets with source IP addresses from your neighbor's network but deny everything else. (the neighbor connected to the other end of the serial interface, so for router 14 this would be router 12's network which is network 100.4.16.0 mask 255.255.240.0)

```
ip access-list extended TestC
permit <neighbor-network> <mask>
deny any any log
```

Which access list would you apply to which interface on a specific router? If you are applying a filter to an inbound interface from your customers, then TestC would be a good practical filter. Apply this list to your router's serial interface as an incoming filter.

```
interface <interface name>
ip access-group TestC in
```

You can also apply either of TestA or TestB as an outbound filter. Note that the former allows all traffic with exception of the addresses that are listed in RFC3330 and are still applicable today whereas the latter allows only traffic with specific source addresses from known subnet(s). TestA is more practical for scenarios where the source addresses can come from a variety of networks.

9. Enable uRPF

Unicast RPF enables a device to verify that the source address of a forwarded packet can be reached through the interface that received the packet. You must not rely on Unicast RPF as the only protection against spoofing. Spoofed packets could enter the network through a Unicast RPF-enabled interface if an appropriate return route to the source IP address exists. Unicast RPF relies on you to enable Cisco Express Forwarding on each device, and is configured on a per-interface basis.

Unicast RPF can be configured in one of two modes: loose or strict. In cases where there is asymmetric routing, loose mode is preferred because strict mode is known to drop packets in these situations. During configuration of the ip verify interface configuration command, the keyword ***any*** configures loose mode while the keyword ***rx*** configures strict mode.

```
Router# config t
Router(config)# ip cef
Router(config)# interface <interface>
Router(config-if)# ip verify unicast source reachable-via <mode>
Router(config-if)# exit
Router(config)# exit
```

10. Configure Sanity Prefix Filters for BGP

Prefix lists allow a network administrator to permit or deny specific prefixes that are sent or received via BGP. Prefix lists should be used where possible to ensure network traffic is sent over the intended paths. Prefix lists should be applied to each eBGP peer in both the inbound and outbound directions.

Configured prefix lists limit the prefixes that are sent or received to those specifically permitted by the routing policy of a network. If this is not feasible due to the large number of prefixes received, a prefix list should be configured to specifically block known bad prefixes. These known bad prefixes include unallocated IP address space and networks that are reserved for internal or testing purposes by RFC 3330. Outbound prefix lists should be configured to specifically permit only the prefixes that an organization intends to advertise.

This configuration example uses prefix lists to ensure that no bogon routes are learned or advertised.

Create the prefix filter named 'bogon-filter':

```
ip prefix-list bogon-filter deny  0.0.0.0/8  le  32
ip prefix-list bogon-filter deny  10.0.0.0/8  le  32
ip prefix-list bogon-filter deny  127.0.0.0/8  le  32
ip prefix-list bogon-filter deny  169.254.0.0/16  le  32
ip prefix-list bogon-filter deny  172.16.0.0/12  le  32
ip prefix-list bogon-filter deny  192.0.2.0/24  le  32
ip prefix-list bogon-filter deny  192.168.0.0/16  le  32
ip prefix-list bogon-filter deny  224.0.0.0/3  le  32
ip prefix-list bogon-filter deny  0.0.0.0/0  le  32
```

Apply this filter to your eBGP neighbor:

```
Router bgp <AS number>
neighbor <neighbor IP address>  remote-as <AS number>
neighbor <neighbor IP address>  version 4
neighbor <neighbor IP address>  prefix-list bogon-filter in
neighbor<neighbor IP address>   prefix-list bogon-filter out
```

Note that it is common for smaller ISPs to have an upstream send only a default route which is enforced via an inbound prefix list. It is also common to create outbound prefix lists to announce only the aggregate allocated prefix outbound to the upstream ISP.

The configuration would be as follows:

```
ip prefix-list DEFAULT-IN seq 5 permit 0.0.0.0/0
ip prefix-list BGP-OUTBOUND seq 5 permit <specific aggregate prefix>
!
router bgp <asn>
 neighbor <ip-address> prefix-list DEFAULT-IN in
 neighbor <ip-address> prefix-list BGP-OUTBOUND out
```

11. Configure MD5 keys on eBGP peers

Peer authentication using MD5 is configured by using the password option to the neighbor BGP router configuration command. The use of this command is illustrated as follows:

```
router bgp <asn>
 neighbor <ip-address> remote-as <remote-asn>
 neighbor <ip-address> password <secret>
```

Note that your routers are already configured with MD5 for your eBGP peers. For this exercise, partner with one of the eBGP or iBGP peers. See what happens when you type in 'no service password-encryption' in configuration mode.:

```
Router# config t
Router(config)# no service password-encryption
Router(conf)# exit
Router# write term
```

Do you see the MD5 passwords? Now type 'service password encryption' again to ensure all passwords in configuration file get encrypted:

```
Router# config t
Router(config)# service password-encryption
Router(conf)# exit
Router# write term
```

Now, look at your routing table (show ip route) and pay careful attention to the routes received from your eBGP or iBGP speaking partner. Configure one side to have a new password. What does your routing table tell you? Can you get any other information from any BGP debug or show commands?

Next, configure the other side to have the same matching password. Check your routing tables and BGP neighbor tables again to ensure that the routing has stabilized again.

LAB III: IPv6 Infrastructure Security

12. Securing the device in an IPv6 network

a. Enable IPv6 and configure interface IP addresses

```
router# config terminal
router(config)# ipv6 unicast-routing
router(config)# interface <interface>
router(config-if)# ipv6 enable
router(config-if)# ipv6 address <ipv6 address>
```

b. Check interface status and neighbor caches to see what it looks like.

```
router# show ipv6 interface
router# show ipv6 neighbors
```

c. Configure a filter to allow only the trusted hosts to have Telnet access. Note that all tries are logged to have an audit trail of all access to the router.

```
router(config)# ipv6 access-list v6_telnet-filter
router(config-ipv6-acl)# permit host <ipv6 address> host <ipv6 address>
```

d. Apply filter to vty ports

```
router(config)# line vty 0 4
router(config-line)# ipv6 access-class v6_telnet-filter in
```

Test to make sure that only telnet from the configured host can have access to the router. Use the debug command to see if you can capture the telnet packets and see the clear-text passwords.

e. Create the filter to allow SSH access. Create a new filter which can be tested and then later the old one can be removed:

```
router(config)# ipv6 access-list v6_ssh-filter
router(config-ipv6-acl)# permit host <ipv6 address> host <ipv6 address>
```

f. Modify vty access command to allow ssh:

```
router(config)# line vty 0 4
router(config-line)# ipv6 access-class v6_ssh-filter in
router(config-line)# transport input ssh
```

Test to make sure that ssh from the permitted host can get access to the router. Use the debug command to see if you can capture the telnet packets and see the encrypted passwords.

13. Securing the data path in an IPv6 network

a. Configure simple traffic filter to announce only your subnet (BCP38)

```
router(config)# ipv6 access-list ipv6-netannounce
router(config-ipv6-acl)# permit ipv6 <v6 subnet> any
router(config-ipv6-acl)# deny ipv6 any any log
router(config-ipv6-acl)# exit
router(config)# interface <interface>
router(config-if)# ipv6 traffic-filter ipv6-netannounce out
```

b. Filter RH Type0 packets

Note: need 12.4(2) T or higher for the following configuration to work.

```
router(config)# ipv6 access-list deny-sourcerouted
router(config-ipv6-acl)# deny ipv6 any any routing-type 0
router(config-ipv6-acl)# permit ipv6 any any
router(config-ipv6-acl)# exit
router(config)# interface <interface>
router(config-if)# ipv6 source-route
router(config-if)# ipv6 traffic-filter deny-sourcerouted in
```

For versions prior to 12.4(2) T, the following configuration will work but it filters all RH Type packets

```
router(config)# no ipv6 source route
```

c. Configure inbound packet filter for initial ipv6 testing

```
router(config)# ipv6 access-list v6starter
router(config-ipv6-acl)# permit icmp any any <ipv6 subnet> echo-reply log-input
router(config-ipv6-acl)# permit icmp any any <ipv6 subnet> echo-request log-input
router(config-ipv6-acl)# permit icmp any any <ipv6 subnet> time-exceeded log-input
router(config-ipv6-acl)# permit icmp any any <ipv6 subnet> packet-too-big log-input
router(config-ipv6-acl)# permit icmp any any <ipv6 subnet parameter-problem log-input
router(config-ipv6-acl)#  permit ipv6 any host <specific host> log-input
router(config-ipv6-acl)# deny ipv6 any any log-input
router(config-ipv6-acl)# exit
router(config)# interface <interface>
router(config-int)# ipv6 traffic-filter v6starter in
```

Note that the 'log-input' is more to check what ipv6 traffic is coming in from the outside. Send some ipv6 pings and see if can see traffic from a 'show log'.

14. Securing the routing infrastructure in an IPv6 network

Repeat lab 10 but using IPv6 addresses using the prefix list named 'ipv6-special-use-pfx':

```
ipv6 prefix-list ipv6-special-use-pfx deny 0::/0 le 128
ipv6 prefix-list ipv6-special-use-pfx deny 0::1/128 le 128
ipv6 prefix-list ipv6-special-use-pfx deny 0::/128
ipv6 prefix-list ipv6-special-use-pfx deny 0::/96
ipv6 prefix-list ipv6-special-use-pfx deny 0::ffff:0:0/96
ipv6 prefix-list ipv6-special-use-pfx deny 0::/8 le 128
ipv6 prefix-list ipv6-special-use-pfx deny fe80::/10 le 128
ipv6 prefix-list ipv6-special-use-pfx deny fc00::/7 le 128
ipv6 prefix-list ipv6-special-use-pfx deny fe00::/9 le 128
ipv6 prefix-list ipv6-special-use-pfx deny ff00::/8 le 128
ipv6 prefix-list ipv6-special-use-pfx deny 2001:db8::/32 le 128
ipv6 prefix-list ipv6-special-use-pfx deny 3ffe::/16 le 128
```

Check your IPv6 routing tables and make sure noone has tried to send you any bogus routing information. [hint: someone might be trying]

LAB IV: Logging/Auditing

15. Setting up Netflow

NetFlow identifies anomalous and security-related network activity by tracking network flows. NetFlow data can be viewed and analyzed via the command line interface (CLI), or the data can be exported to a commercial or freeware NetFlow collector for aggregation and analysis. NetFlow collectors, through long-term trending, can provide network behavior and usage analysis. NetFlow functions by performing analysis on specific attributes within IP packets and creating flows. Version 5 is the most commonly used version of NetFlow, however, version 9 is more extensible. NetFlow flows can be created using sampled traffic data in high-volume environments.

Cisco Express Forwarding (CEF), or distributed CEF, is a prerequisite to enabling NetFlow. NetFlow can be configured on routers and switches.

This example illustrates the basic configuration of this feature. In previous releases of Cisco IOS software, the command to enable NetFlow on an interface is *ip route-cache flow* instead of *ip flow {ingress | egress}*.

```
Router(config)# ip flow-export destination <ip-address> <udp-port>
Router(config)# ip flow-export version <version>
Router(config)# interface <interface>
Router(config-if)# ip flow <ingess|egress>
```

The following is an example of NetFlow output from the CLI. The SrcIf attribute can aid in traceback.

```
router#show ip cache flow
IP packet size distribution (26662860 total packets):
   1-32   64   96  128  160  192  224  256  288  320  352  384  416  448  480
   .741 .124 .047 .006 .005 .005 .002 .008 .000 .000 .003 .000 .001 .000 .000

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .000 .000 .001 .007  .039  .000  .000 .000  .000  .000   .000

IP Flow Switching Cache, 4456704 bytes
  55 active, 65481 inactive, 1014683 added
  41000680 ager polls, 0 flow alloc failures
  Active flows timeout in 2 minutes
  Inactive flows timeout in 60 seconds
IP Sub Flow Cache, 336520 bytes
  110 active, 16274 inactive, 2029366 added, 1014683 added to flow
  0 alloc failures, 0 force free
  1 chunk, 15 chunks added
  last clearing of statistics never
Protocol         Total    Flows   Packets Bytes  Packets Active(Sec) Idle(Sec)
--------         Flows     /Sec     /Flow  /Pkt     /Sec     /Flow     /Flow
TCP-Telnet       11512      0.0        15    42      0.2      33.8      44.8
TCP-FTP           5606      0.0         3    45      0.0      59.5      47.1
TCP-FTPD          1075      0.0        13    52      0.0       1.2      61.1
TCP-WWW          77155      0.0        11   530      1.0      13.9      31.5
TCP-SMTP          8913      0.0         2    43      0.0      74.2      44.4
TCP-X              351      0.0         2    40      0.0       0.0      60.8
TCP-BGP            114      0.0         1    40      0.0       0.0      62.4
TCP-NNTP           120      0.0         1    42      0.0       0.7      61.4
TCP-other       556070      0.6         8   318      6.0       8.2      38.3
UDP-DNS         130909      0.1         2    55      0.3      24.0      53.1
UDP-NTP         116213      0.1         1    75      0.1       5.0      58.6
UDP-TFTP           169      0.0         3    51      0.0      15.3      64.2
UDP-Frag             1      0.0         1  1405      0.0       0.0      86.8
UDP-other        86247      0.1       226    29     24.0      31.4      54.3
ICMP             19989      0.0        37    33      0.9      26.0      53.9
IP-other           193      0.0         1    22      0.0       3.0      78.2
Total:         1014637      1.2        26    99     32.8      13.8      43.9

SrcIf     SrcIPaddress     DstIf     DstIPaddress     Pr   SrcP   DstP   Pkts
Gi0/1     192.168.128.21   Local     192.168.128.20   11   CB2B   07AF      3
Gi0/1     192.168.150.60   Gi0/0     10.89.17.146     06   0016   101F     55
Gi0/0     10.89.17.146     Gi0/1     192.168.150.60   06   101F   0016      9
Gi0/1     192.168.150.60   Local     192.168.206.20   01   0000   0303     11
Gi0/0     10.89.17.146     Gi0/1     192.168.150.60   06   07F1   0016      1
```